Polynomial Time Scheduling of Low Level Computer Vision Algorithms on Networks of Heterogeneous Machines

Adam R. Nolan , anolan@ece.uc.edu
Bryan Everding, beverdin@ece.uc.edu
Artificial Intelligence and Computer Vision Lab
University of Cincinnati †

Abstract

Defining an optimal schedule for arbitrary algorithms on a network of heterogeneous machines is an NP complete problem. This paper focuses on data parallel deterministic neighborhood computer vision algorithms. This focus enables the polynomial time definition of a schedule which minimizes the distributed execution time by overlapping computation and communication cycles on the network. The scheduling model allows for any speed machine to participate in the concurrent computation but makes the assumption of a master/slave control mechanism using a linear communication network. Several vision algorithms are presented and described in terms of the scheduling model parameters. The theoretical speedup of these algorithms is discussed and empirical data is presented and compared to theoretical results.

Keywords: Computer Vision, Heterogeneous Architectures, Scheduling, Distributed Algorithms

1 Introduction

In the past many studies have been performed analyzing the capabilities of various parallel processor – vision algorithm mappings. Thorough surveys can be found in [18][3][2][4][5][17]. Most of these efforts focus on the mapping of a single machine to a single algorithm or mapping a suite of algorithms to a single architecture. [18] Most of the conclusions made in these studies are based on the architectural similarities between the hardware communication configurations and the communication patterns inherent in the vision algorithm (i.e. vision tasks tend to have highly regular communication). Recent research efforts have discussed the mapping of computer vision tasks to networks of workstations with the assumption of homogeneous workstation clusters. [11][7] Additional efforts have focused on scheduling suites of independent programs onto networks of heterogeneous machines. [10] This paper relaxes the assumption of homogeneous workstation clusters and independent program suites. It focuses on the distribution of a single program on a set of architectures connected by the PVM message passing library. [6][13] A framework is presented for the polynomial time scheduling of deterministic local communication algorithms onto a suite of heterogeneous machines using linear communication.

The paper is organized as follows. Section 2 presents the background and motivation. Section 3 develops the analytical description of the scheduling process and introduces a set of conditions necessary for the minimization of the execution time. Section 4 presents the scheduling algorithm. Section 5 presents several low level computer vision tasks and their corresponding scheduling models. Section 6 demonstrates the use of the scheduling method on the computer vision examples, presents theoretical speedups and discusses the elements of nondeterminism inherent in actual run times.

†This work was supported by the NASA – University of Cincinnati Space Engineering Research Center

Section 7 presents several conclusions based on the experimental data, and finally an appendix contains derivations of several conditions presented in the paper.

2 Background

Efforts have been made to develop an Automatic Visual Inspection System (AVIS) for use with various critical aerospace components. These components include high pressure turbine blades (aircraft), injector baffles, oxidation posts, annulus rings(spacecraft). AVIS utilizes several scales of information abstracted from the original image, with each scale requiring a set of low level vision operations. [8][14] The realization of the AVIS paradigm is limited by the tremendous computational burden of these low level vision operations. These algorithms include convolution, difference of Gaussian filtering, morphological filtering, Fourier transform, and Hough transform. In order to increase the speed of AVIS, distributed solutions were investigated. Defining an effective distribution onto the various machines available on the LAN requires models of algorithm decomposition, communication mechanisms, and machine speed for a given algorithm.

Heterogeneous computing is the well orchestrated and coordinated effective use of a suite of diverse high-performance machines to provide superspeed processing. [10] These applications typically contain several types of distinct control and data parallelism. For the purposes of this work data parallelism is of primary interest. Of the available models of parallelism the master/slave paradigm is efficient for most low level computer vision algorithms. The master-slave model utilizes a single controlling machine which issues commands and data to slave machines. In data-parallelism these commands are identical typically operating on different sets of data (the Single Program Multiple Data model-SPMD) or occasionally on identical data sets in the case of distributed search methods. Whereas in control-parallelism the slave machines perform dissimilar tasks on typically dissimilar data sets. Most low level vision tasks exhibit data parallelism with deterministic and localized communication patterns. This characteristic enables the elimination of interslave communication by sending each slave machine the necessary data according to the algorithm neighborhood. As the major computational burden of AVIS corresponds to low level vision algorithms, these are pursued in depth in the upcoming sections.

3 Analytical Development

Before an algorithm can be scheduled a decomposition scheme must be defined for the data. Two popular methods for decomposing image data are row partitioning (or strip mining) and block partitioning. In row partitioning each slave processor receives a number of rows from the original image and the associated neighborhood pixels needed in order to manipulate this row of data. Likewise, in block partitioning each machine receives a given block of data and its associated neighborhood. Typically these blocks must be partitioned onto a square number of homogenous machines, although alternative heuristic schemes have been described for blocking nonsquare numbers of homogeneous machines. [11] Examples of these partitioning schemes can be seen in Fig.1 & Fig.2. Although the total boundary pixels are typically less for block partitioned data than for row partitioned data, $\left(2\sqrt{n}-2\right)$ vs $(n-1)$,

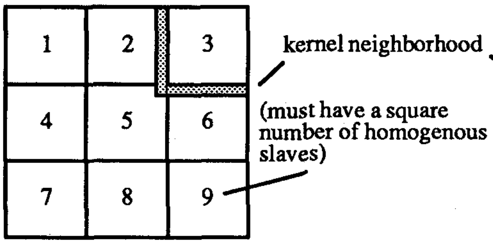


Fig.1 Block partitioning– 9 machines

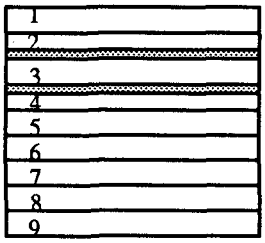


Fig.2 Row Partitioning – 9 machines

block partitioning introduces many problems in the scheduling of heterogeneous machines. Assuming that an efficient heuristic could be defined that would partition the data onto heterogeneous machines, its inherent nonlinearity would make the analysis and minimization computationally expensive (at least combinatorial in regards to the number of nonlinearities). Hence, row partitioning will be assumed for the upcoming formulations. In order to discuss the analytic model and the conditions necessary for optimal scheduling, terminology involved with the theoretic execution and communication times must be presented. Definitions and terms are presented in Fig.3 .

Definitions:

$\mathcal{S}(n)$ – a scheduling of N homogeneous tasks onto n machines, where

$N = \sum_{i=1}^{i=n} \eta_i$, with each slave machine(i) receiving a task size of η_i .

γ_i – seconds/operation for each processor(i), where the operation is algorithm dependent.

$T^y_{r(i)}$ – time(secs) spent receiving data from y tasks from processor i .

$T^0_{r(i)}$ – overhead(secs) required for any receive operation .

$T_{r(i)}$ – time(secs) spent receiving data from η_i tasks from processor i .
$T_{r(i)} = T^0_{r(i)} + T^{\eta_i}_{r(i)}$

ϕ_r – #bytes needed to receive task size 1 from a slave machine .

$T^y_{s(i)}$ – time(secs) spent sending data associated with y tasks to processor i .

$T^0_{s(i)}$ – overhead(secs) required for any send operation .

$T_{s(i)}$ – time(secs) spent sending data for η_i tasks to processor i .
$T_{s(i)} = T^0_{s(i)} + T^{\eta_i}_{s(i)}$

ϕ_s – #bytes needed to send task size 1 to a slave machine .

$T^y_{c(i)}$ – time(secs) required by processor i to compute a task size of y .

$T_{c(i)}$ – time(secs) spent by processor i computing task size η_i , $T_{c(i)} = T^{\eta_i}_{c(i)}$.

ϕ_c – #operations required to compute a task of size 1 .

T(n) – total time to execute $\mathcal{S}(n)$.

$T^{a-b}_{c(i)} = T^a_{c(i)} - T^b_{c(i)}$.

$\lfloor f \rfloor = \max(0, f)$.

Fig.3 Definitions and terminology used in the scheduling model

For the following discussions, a linear single line communication model is assumed. Hence the communication time can be modeled as $T_{communiation} = \alpha + \#bytes \times \beta$, where α is the startup overhead for a particular communication process and β is the seconds per byte transmission rate of data to and from a slave processor. This results in a constant aggregate communication time for a given number of processors: $\sum_{i=1}^{n} T_{s(i)} = T_s$, $\sum_{i=1}^{n} T_{r(i)} = T_r$. Similarly, the addition of any new slave machines into the data parallel non interslave communication paradigm results in an additional overhead value: $\sum_{i=1}^{n+1} T_{s(i)} = \sum_{i=1}^{n} T_{s(i)} + T^0_{s(n+1)}$ and similarly: $\sum_{i=1}^{n+1} T_{r(i)} = \sum_{i=1}^{n} T_{r(i)} + T^0_{r(n+1)}$. Some additional assumptions included in the AVIS model are that the slave machines can be organized such that machine(i) is faster than machine(i+1), i.e. $\gamma_i < \gamma_{i+1}$ for the particular type of task set to be scheduled. Also assumed is that the time for communication is not greater than the time for computation, i.e. $T_{c(i)} > T_{s(i)} + T_{r(i)} \ \forall\, i$. These assumptions are necessary for several of the analytic derivations.

As seen in Fig.4 the linear communication model allows for a quick graphical analysis of the optimality of a given schedule. The degree of nonconcurrency in the

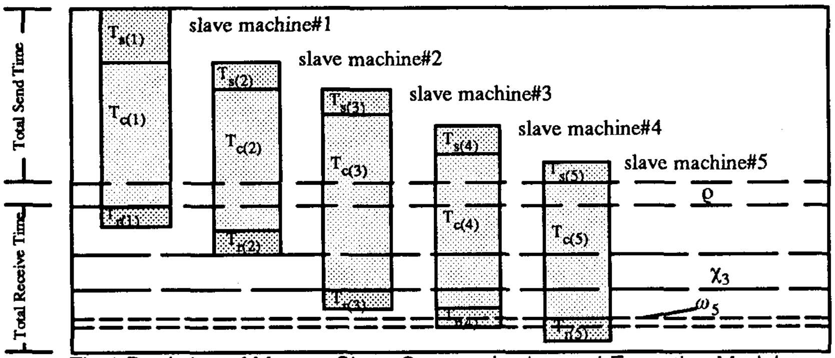


Fig.4 Depiction of Master–Slave Communication and Execution Model

schedule is indicated by the values of ϱ, ω, and χ. The parameter ϱ is a measure of the time spent by the master machine waiting for the first processed data partition to be returned, $\varrho = T_{c(1)} - \sum_{i=2}^{n} T_{s(i)}$. The parameter ω_i is a measure of slave machine wait time caused by contention with the previous slave machine for the linear communication channel, $\omega_i = \lfloor T_{c(i-1)} + T_{r(i-1)} - \left(T_{s(i)} + T_{c(i)} \right) \rfloor$. The parameter χ_i is a measure of the master wait time caused by excess computation of machine(i) after machine(i–1) has completed returning its data partition. Although it would appear that $\chi_i = \lfloor T_{s(i)} + T_{c(i)} - T_{c(i-1)} - T_{r(i-1)} \rfloor$, χ_i can be reduced by previous ω_j values, i.e. $\chi_i = \lfloor T_{c(i)} - \sum_{j=i+1}^{n} T_{s(j)} - \sum_{j=1}^{i-1} T_{r(j)} - \varrho \rfloor$. As one would expect, large values of ϱ, ω and χ result in suboptimal schedules. Typically these values cannot be eliminated due to the constraint of integer task size. The total execution time for a given schedule is:

$T(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \lfloor \varrho \rfloor + \sum_{i=2}^{n} \chi_i$. In order to minimize this execution time, a set of criteria must be established to explicitly define the effects of the values ϱ, ω and χ. The following conditions are used to establish limits on the relative sizes of these parameters.(Derivations can be found in the Appendix.)

Condition 1:

For any given schedule $\mathcal{S}(n)$ in which $\chi_j \geq T^1_{c(j)} + T^1_{c(1)}$, $j > 1$, there exists an alternative schedule $\tilde{\mathcal{S}}(n)$ with $\tilde{\chi}_j < T^1_{c(j)} + T^1_{c(1)}$ such that $\tilde{T}(n) < T(n)$. Hence, from this criterion one can conclude that a schedule must have $\chi_j < T^1_{c(j)} + T^1_{c(1)}$, $j > 1$, otherwise the schedule will be suboptimal.

Condition 2:

Given a schedule $\mathcal{S}(n)$ with $\omega_j \geq T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$ for some j, there exists an alternative schedule $\tilde{\mathcal{S}}(n)$ with $\omega_j < T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$ such that $\tilde{T}(n) \leq T(n)$.
From this conclusion, it follows that any minimum timschedule must have $\omega_j < T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$.

Condition 3:

For any given schedule $\mathcal{S}(n)$ satifying conditions 1&2 with $\varrho > T^0_{s(n+1)} + T^0_{r(n+1)}$, there exists an alternative schedule $\mathcal{S}(n+1)$ such that $T(n+1) < T(n)$.
This presents a condition to indicate when the number of slave machines should be increased. (Fig.5)

Condition 4:

For any given schedule $\mathcal{S}(n)$ satisfying conditions 1&2 such that $\varrho < T^0_{r(n)} + T^{\tilde{\eta}_1-\eta_1}_{s(1)} - T^{\tilde{\eta}_1-\eta_1}_{c(1)}$ there exists an alternative schedule $\mathcal{S}(n-1)$ such that $T(n-1) < T(n)$, where $\tilde{\eta}_1$ corresponds to the alternate schedule $\mathcal{S}(n-1)$ as defined below in (*). This presents a condition to indicate when the number of slav machines should be decreased. (Fig.6)

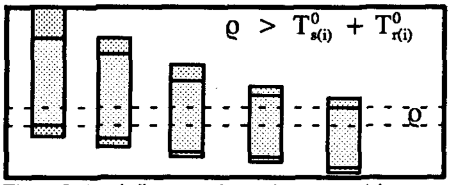
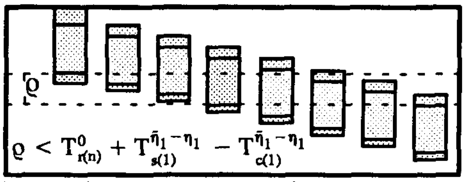


Fig.5 Scheduling too few slave machines



ϱ
$\varrho < T^0_{r(n)} + T^{\tilde{\eta}_1-\eta_1}_{s(1)} - T^{\tilde{\eta}_1-\eta_1}_{c(1)}$

Fig.6 Scheduling too many slave machines

4 Scheduling Algorithm

Relaxing the integral constraint on the scheduled task size, a minimal time schedule (for n machines) can be found in which ω_i=0, χ_i=0. This solution corresponds to $T_{c(i)} + T_{r(i)} = T_{c(i+1)} + T_{s(i+1)}$. Using this relation in conjunction with the definition of total task size, $\sum_{i=1}^{i=n} \eta_i = N$, the n values of η_i can be determined explicitly using Gaussian elimination on the n linear equations shown in Fig.7 .

$$\begin{array}{ll} (1) & T_{c(1)} + T_{r(1)} = T_{c(2)} + T_{s(2)} \\ \vdots & \\ (n-1) & T_{c(n-1)} + T_{r(n-1)} = T_{c(n)} + T_{s(n)} \\ (n) & \sum_{i=1}^{n} \eta_i = N \end{array}$$

Fig.7 Set of linear equations defining schedule with $\chi_i = 0$, $\omega_i = 0$ for all i

Although no closed form solution for n exists, an appropriate number of slave machines can be determined by the evaluation of ϱ as described by conditions 3&4. The computational cost of finding the appropriate number of machines is very low due to the assumption of linear communication. As $\varrho = T_{c(1)} - \sum_{i=2}^{n} T_{s(i)}$, a change in the number of slave machines requires only the recalculation of η_1 in order to evaluate the new ϱ value. Constructing the linear system of equations as:

$$\begin{array}{ll} (1) & \eta_2 = a_2\eta_1 + b_2 \\ (2) & \eta_3 = a_3\eta_1 + b_3 \\ \vdots & \\ (i) & \eta_i = a_i\eta_1 + b_i \\ \vdots & \\ (n-1) & \eta_n = a_n\eta_1 + b_n \end{array}$$

$$\text{where } a_i = \left(\frac{\gamma_{i-1}\phi_c + \beta\phi_r}{\gamma_i\phi_c + \beta\phi_s}\right) a_{i-1} \text{ for } i > 1,$$

$$b_i = \frac{T^0_{r(i-1)} - T^0_{s(i)}}{\gamma_i\phi_c + \beta\phi_s} + b_{i-1} \text{ for } i > 1,$$

$$b_1 = 0,\ a_1 = 1 .$$

We can calculate η_1 directly for any n (after the a_i and b_i values are found) using:

$$\eta_1 = \frac{N - \sum_{i=1}^{n} b_i}{\sum_{i=1}^{n} a_i} \qquad (*)$$

Hence the algorithm requires 2p steps to construct the a_i and b_i values, a maximum of p steps to define an appropriate number of slave machines, and a maximum of p steps to define the η_i values for the schedule, where p is the total number of slave machines.

As an actual schedule must stick to integer task sizes, the real valued task sizes must be converted to integral values in a manner which does not violate conditions 1,2,3,4. This integer approximation, $\bar{\eta}_i$, must also satisfy the condition: $\sum_{i=1}^{n} \bar{\eta}_i = N$. A histogram based thresholding technique is used on the non–integral portions of the η_i values. This procedure selects a threshold ζ such that $\sum_{i=1}^{n} \bar{\eta}_i = N$ and requires kn steps where k is the number of bins used in the histogram. As shown in the Appendix, this thresholding method does not violate conditions 1 or 2.

5 Low Level Vision Modeling

The low level computer vision tasks needed for the AVIS computations are: convolution, difference of Gaussian, Fourier transform, Hough transform, and morphological filtering operations. Although explication of these of these algorithms is beyond the scope of this paper, thorough discussions can be found in [1][4][5][12][16][17][18]. For the purpose of this study, the key algorithmic elements are contained in the model parameters: $\gamma_i, \phi_c, \phi_s, \phi_c, T_s^0, T_r^0$. These algorithm specific parameters are presented in Fig.8.

	Convolution	DoG	Morphological	2D FFT	Hough
γ_i	$\frac{\text{secs}}{\text{mult + add}}$	$\frac{\text{secs}}{\text{mult + add}}$	$\frac{\text{secs}}{\text{2logical Ops}}$	$\frac{\text{secs}}{\text{cmplx(mult + add)}}$	$\frac{\text{secs}}{\text{mult + trig\_lookup}}$
ϕ_c	M^2N	$(M_1^2 + M_2^2)N$	BN	$N\log_2 N$	1
ϕ_s	N	N	N	N	0
ϕ_r	N	N	N	N	1
$T_{s(i)}^0$	$\alpha + (M^2 + (M-1)N)\beta$	$\alpha + (M_1^2 + M_2^2 + (M_1 - 1)N)\beta$	$\alpha + (B + (M-1)N)\beta$	α	$\alpha + N^2\beta$
$T_{r(i)}^0$	α	α	α	α	α

Fig.8 Table of scheduling parameters for the low level vision algorithms.(α is the communication channel overhead (sec), β is the sec/byte transfer rate of the communication channel, the image is size NxN, the kernel(i) size is M_ixM_i, and the number of elements in the morphological kernel is B.)

6 Experimental Results

The heterogeneous distributed architecture included several machine types and configurations. The machines included HP 9000 715/33s, a MasPar MP–1(1024 processors), and SUN SPARCstations– IPXs, LXs, IIGXs, 5s, 20s(1&4processor). For a given algorithm, a testing set was generated and run on each architecture. Based on the results of the test suite, a single γ value was assigned to each architecture for that algorithm type. The sample mean values of the experimental gamma distributions were used for this single value. These values were stored and selectively loaded depending on the configuration of the parallel virtual machine being used. Similarly, the sample means of the α and β values were used in the scheduling experiments(8000μs&19μs/pixel for the XDR protocol). The scheduling algorithm is depicted in the graphic user interface (Fig.9).

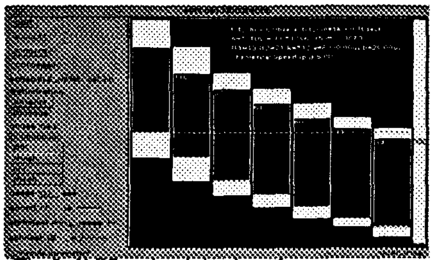


Fig.9 AVIS scheduler interface

Comparison of theoretical and experimental speedups of the modeled schedules are shown in Fig.10&Fig.11 for the convolution and DOG algorithms. As one would expect, the observed speedups were lower than the theoretical. The primary cause for this degradation was the nondeterminism of the communication and execution rates on the various machines. As the current scheduling algorithm uses only sample mean approximations of these nondeterministic values, it is susceptible to variances in these parameters. Efforts were made to run experiments during low usage periods in the day in order to minimize these effects. However, even in an isolated distributed system nondeterminism is evidenced due to memory coherency, operating system overhead, and communication protocols. Discussion of these nondeterministic issues can be found in [15].

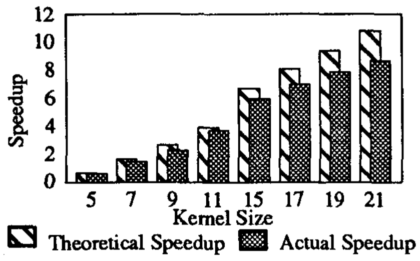


Fig.10 Theoretical vs Actual Speedup of Convolution

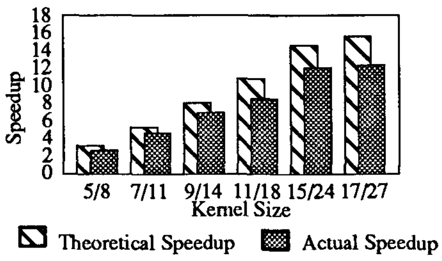


Fig.11 Theoretical vs Actual Speedup of DOG

7 Conclusions

The AVIS GUI Scheduler enables the visualization of the speedup of a given distributed algorithm. The scheduling technique presented enables the generation of a minimum time schedule in polynomial time. The relaxation on machine architecture homogeneity enables efficient solutions to low level vision problems using any set of Unix machines one has at their disposal. This makes for a system which is both cost effective and reconfigurable.

Analyses were presented in terms of machine specific $T^0_{s(i)}$, and $T^0_{r(i)}$ values. These enable a precise description of specific slave communication overheads. Similarly, the definition of machine specific neighborhood functions($\phi_{c(i)}$, $\phi_{s(i)}$, $\phi_{r(i)}$) is a simple extension of the current algorithm and may allow for a more precise description of architectural differences between platforms (i.e. memory configuration, pipeline depth, vectorization) and communication protocols(XDR, NFS, AFS). Although the formulations presented assume a single β value, they also can be extended to machine specific β_i. Defining β_i enables non–local machines to participate in the distributed algorithm more effectively than the current model allows. It is hoped that the inclusion of these parameters will reduce the parameter variance. As the parameter variance is the primarily cause for error in the predicted speedup of the schedule, this should create a more accurate theoretical model.

Acknowledgements

The authors would like to thank the reviewers for their time and helpful suggestions.

References

[1] Computer Vision, Dana H. Ballard, Christopher M. Brown, Prentice–Hall, Englewood Cliffs, New Jersey, 1982.

[2] Parallel Algorithms for Machine Intelligence and Vision(Vipin Kumar Editor) ,"Parallelism in Computer Vision: A Review", Vipin Chaudhary, J. K. Aggarwal, Springer–Verlag, NY 1990.

[3] Parallel Architectures and Parallel Algorithms for Integrated Vision Systems, Alok N. Choudhary, Janak H. Patel, The Kluwer International Series In Engineering and Computer Science, Kluwer Academic Publishers Boston, 1990.

[4] "SIMD Architectures and Algorithms for Image Processing and Computer Vision", Robert Cypher and Jorge L. C Sanz, *IEEE Trans on ASSP*, Vol. 37, No. 12, Dec. 1989.

[5] "Parallel Algorithms for Low Level Vision on the Homogeneous Multiprocessor", R.V Dantu, N.J Dimopoulos, K.R. Li, R.V Patel, A.J. Al–Khalili, *Computers Elect. Eng*, Vol. 20 No. 1, pp51–60, 1994.

[6] Jack Dongarra, PVM3.3 Users Guide, 1994.
[7] "Parallel Programming Systems For Workstation Clusters", Yale University Department of Computer Science Research Report, YALEU/DCS/TR-975, August, 1993.
[8]"Generalizationof an Automated Visual Inspection System(AVIS)", Bryan S. Everding, Adam R. Nolan, William G. Wee, SAE Aerospace Atlantic Conference, April 1994.
[9] "Heterogeneous Processing", Richard F. Freund, Howard Jay Siegel,Computer, June 1993, p 13.
[10] "The Challenges of Heterogeneous Computing", Richard F. Freund, Proceedings of the Parallel Systems Fair, Cancun, April 1994.
[11] "Efficient Parallel Image Processing Applications On a Network Of Distributed Workstations", Chi-kin Lee, Mounir Hamdi, *Proceedings of the Parallel Systems Fair*, Cancun, April 1994.
[12] Vision, A Computational Investigation into the Human Representation and Processing of Visual Information, David Marr, W.H Freeman Press, San Francisco, 1982.
[13] "LINDA and PVM: A comparison between two environments for parallel programming", Alfonso Matrone, Pasquale Schiano, Vittorio Puoti, Parallel Computing 19(1993) 949-957.
[14]"Performance Issues of an Automated Visual Inspection System(AVIS)", Bryan S. Everding, Adam R. Nolan, William G. Wee, AIAA Jet Propulsion Conference, June 1994
[15] "Effects of Nondeterminism on the Predicted Speedup of Scheduling Low Level Computer Vision Algorithms on Networks of Heterogeneous Machines", A. R. Nolan, B. Everding, W. G. Wee, 5th International Conference on Parallel Computing, Belgium, September, 1995.
[16] Digital Signal Processing. Oppenheim, A.V., and Schafer, R.W., Englewood Cliffs, NJ, Prentice-Hall,. 1975.
[17] "Mapping Vision Algorithms to Parallel Architectures", Quentin Stout, *Proceeding of the IEEE*, Vol. 76. No. 8. August, 1988.
[18] "DARPA Image Understanding Benchmark for Parallel Computers", Charles Weems, Edward Riseman, Allen Hanson, Azriel Rosenfeld, *Journal of Parallel and Distributed Computing*, Vol. 11 No. 1 Jan. 1991, pp1-24.

Appendix

Condition1: For any given schedule $\mathcal{S}(n)$ in which $\chi_j \geq T^1_{c(j)} + T^1_{c(1)}$, $j > 1$,
there exists an alternative schedule $\tilde{\mathcal{S}}(n)$ with $\tilde{\chi}_j < T^1_{c(j)} + T^1_{c(1)}$ such that $\tilde{T}(n) < T(n)$.

Proof: For a given χ_j, there exists a corresponding task size η_{χ_j} defined as:

$$\eta_{\chi_j} = \mathrm{int}_{\min}\left(\frac{\chi_j}{T^1_{c(j)} + T^1_{c(1)}}\right),\quad \tilde{\chi}_j = \chi_j - T^{\eta_{\chi_j}}_{c(1)} - T^{\eta_{\chi_j}}_{c(j)} \text{ which results}$$

in the following bound: $0 \leq \tilde{\chi}_j < T^1_{c(j)} + T^1_{c(1)}$.

Case 1) If $\chi_i = 0 \ \forall \ i < j$ in a given schedule $\mathcal{S}(n)$, the corresponding execution time is:

$$T(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \lfloor \varrho \rfloor + \chi_j + \sum_{i=j+1}^{n} \chi_i$$

defining a new schedule $\tilde{\mathcal{S}}(n)$ such that $\tilde{\eta}_1 = \eta_1 + \eta_{\chi_j}$, $\tilde{\eta}_j = \eta_j - \eta_{\chi_j}$.

$$\tilde{\chi}_j = \chi_j - T^{\eta_{\chi_j}}_{c(1)} - T^{\eta_{\chi_j}}_{c(j)}$$

The execution time for that schedule is $\tilde{T}(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \tilde{\varrho} + \tilde{\chi}_j + \sum_{i=j+1}^{n} \chi_i$

where $\tilde{\varrho} = \varrho + T^{\eta\chi_j}_{c(1)} + T^{\eta\chi_j}_{s(1)}$

Hence, $\tilde{T}(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \varrho + T^{\eta\chi_j}_{c(1)} + T^{\eta\chi_j}_{s(1)} + \tilde{\chi}_j + \sum_{i=j+1}^{n} \chi_i$

If $\varrho \geq 0$, $T(n) - \tilde{T}(n) = \chi_j - T^{\eta\chi_j}_{c(1)} - T^{\eta\chi_j}_{s(1)} - \tilde{\chi}_j = T^{\eta\chi_j}_{c(j)} - T^{\eta\chi_j}_{s(1)} > 0,$

by assumption of $T_c > T_r + T_s$

If $\varrho < 0$, $T(n) - \tilde{T}(n) = \chi_j - \left\lfloor \varrho + T^{\eta\chi_j}_{c(1)} + T^{\eta\chi_j}_{s(1)} \right\rfloor - \tilde{\chi}_j$

$= T^{\eta\chi_j}_{c(j)} + T^{\eta\chi_j}_{c(1)} - \left\lfloor \varrho + T^{\eta\chi_j}_{c(1)} + T^{\eta\chi_j}_{s(1)} \right\rfloor$

$\min\left(T(n) - \tilde{T}(n)\right) = T^{\eta\chi_j}_{c(i)} - T^{\eta\chi_j}_{s(1)} > 0$, $\max\left(T(n) - \tilde{T}(n)\right) = T^{\eta\chi_j}_{c(i)} + T^{\eta\chi_j}_{c(1)} > 0$

Case 2) If $\chi_i > 0$ for some $i < j$ in a given schedule $\mathcal{S}(n)$:

$$\chi_i = \left\lfloor T_{c(i)} - \left(\sum_{k=i+1}^{n} T_{s(k)} + \sum_{k=1}^{i-1} T_{r(k)} + \lfloor \varrho \rfloor \right) \right\rfloor$$

for $i < j$, $$\tilde{\chi}_i = \left\lfloor T_{c(i)} - \left(\sum_{k=i+1}^{n} T_{s(k)} - T^{\eta\chi_j}_{s(j)} + \sum_{k=1}^{i-1} T_{r(k)} + T^{\eta\chi_j}_{r(1)} + \lfloor \tilde{\varrho} \rfloor \right) \right\rfloor$$

as $\tilde{\varrho} = \varrho + T^{\eta\chi_j}_{c(1)} + T^{\eta\chi_j}_{s(j)}$, $\tilde{\chi}_i = \chi_i - \left\lfloor \varrho - T^{\eta\chi_j}_{r(1)} + T^{\eta\chi_j}_{c(1)} \right\rfloor$, $\tilde{\chi}_i \leq \chi_i$

$T(n) - \tilde{T}(n) = \chi_j - T^{\eta\chi_j}_{c(i)} - \tilde{\chi}_j + \chi_i - \lfloor \tilde{\chi}_i \rfloor > 0$

Condition 2 : Given a schedule $\mathcal{S}(n)$ with $\omega_j \geq T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$ for some j , there exists an alternative schedule $\tilde{\mathcal{S}}(n)$ with $\omega_j < T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$ such that $\tilde{T}(n) < T(n)$.

Proof : For any given $\omega_j \geq T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}$ an alternate mapping can be defined as : $\tilde{\eta}_{j-1} = \eta_{j-1} - \eta_{\omega_j}$, $\tilde{\eta}_j = \eta_j + \eta_{\omega_j}$

where $$\eta_{\omega_j} = \mathrm{int}_{\min}\left(\frac{\omega_j}{T^1_{c(j-1)} + T^1_{r(j-1)} + T^1_{c(j)} + T^1_{s(j)}} \right)$$

Case 1) $j = 2$, $\tilde{\varrho} = \varrho - T^{\eta\omega_j}_{c(1)} - T^{\eta\omega_j}_{s(1)}$,

If $\varrho \geq T^{\eta\omega_j}_{c(1)} - T^{\eta\omega_j}_{s(1)}$, $T(n) - \tilde{T}(n) = T^{\eta\omega_j}_{c(1)} + T^{\eta\omega_j}_{s(1)} > 0$.

If $\varrho < T^{\eta\omega_j}_{c(1)} - T^{\eta\omega_j}_{s(1)}$, $T(n) - \tilde{T}(n) = \varrho \geq 0$.

Case 2) $j > 2,\ \tilde{\chi}_{j-1} = \chi_{j-1} - T^{\eta\omega_j}{}_{c(j-1)} - T^{\eta\omega_j}{}_{s(j-1)}$

$$T(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \lfloor \varrho \rfloor + \sum_{i=2}^{n} \chi_i$$

$$\tilde{T}(n) = \sum_{i=1}^{n} T_{s(i)} + \sum_{i=1}^{n} T_{r(i)} + \lfloor \varrho \rfloor + \sum_{i=2}^{j-2} \chi_i + \tilde{\chi}_{j-1} + \tilde{\chi}_j + \sum_{i=j+1}^{n} \chi_i,$$

As $\tilde{\chi}_j \leq 0$, If $\tilde{\chi}_{j-1} \geq 0,\ T(n) - \tilde{T}(n) = T^{\eta\omega_j}{}_{c(j-1)} + T^{\eta\omega_j}{}_{s(j)} > 0$.

If $\tilde{\chi}_{j-1} < 0,\ T(n) - \tilde{T}(n) = \chi_{j-1} \geq 0$.

Condition 3: For any given schedule $\mathcal{S}(n)$ with $\varrho > T^0_{s(n+1)} + T^0_{r(n+1)}$, satifying conditions 1&2 there exists an alternative schedule $\mathcal{S}(n+1)$ such that $T(n+1) < T(n)$.

Proof : Define a new mapping $\mathcal{S}(n+1)$ such that $\tilde{\eta}_1 = \dfrac{N - \sum_{i=1}^{n+1} b_i}{\sum_{i=1}^{n+1} a_i}$ as defined in (*)

The schedule $\mathcal{S}(n+1)$ has an execution time of $T(n+1) = \sum_{i=1}^{i=n+1} T_{s(i)} + \sum_{i=1}^{i=n+1} T_{r(i)} + \lfloor \tilde{\varrho} \rfloor$

$$T(n+1) = \sum_{i=1}^{i=n} T_{s(i)} + T^0_{s(n+1)} + \sum_{i=1}^{i=n} T_{r(i)} + T^0_{r(n+1)} + \lfloor \tilde{\varrho} \rfloor$$

$$\tilde{\varrho} = \left\lfloor T^{\tilde{\eta}_1}_{c(1)} - \sum_{i=2}^{n} T_{s(i)} - T^0_{s(n+1)} - T^{\eta_1 - \tilde{\eta}_1}_{s(1)} \right\rfloor$$

$$T(n) - T(n+1) = \varrho - \lfloor \tilde{\varrho} \rfloor + \sum_{i=1}^{i=n} T_{s(i)} + \sum_{i=1}^{i=n} T_{r(i)} - \sum_{i=1}^{i=n+1} T_{s(i)} - \sum_{i=1}^{i=n+1} T_{r(i)}$$

Case1) $\tilde{\varrho} \geq 0$,$T(n) - T(n+1) = T^{\eta_1 - \tilde{\eta}_1}_{c(1)} + T^{\eta_1 - \tilde{\eta}_1}_{s(1)} - T^0_{r(n+1)} > 0$.

Case2) $\tilde{\varrho} < 0$,$T(n) - T(n+1) = \varrho - T^0_{r(n+1)} - T^0_{s(n+1)} > 0$.

Condition 4: For any given schedule $\mathcal{S}(n)$ with $\varrho < T^0_{r(n)} + T^{\tilde{\eta}_1 - \eta_1}_{s(1)} - T^{\tilde{\eta}_1 - \eta_1}_{c(1)}$ satisfying conditions1&2, there exists an alternative schedule $\mathcal{S}(n-1)$ such that $T(n-1) < T(n)$.

Proof : $\mathcal{S}(n)$ with the above constraints results in $T(n) = \sum_{i=1}^{i=n} T_{s(i)} + \sum_{i=1}^{i=n} T_{r(i)}$

Define a new mapping $\mathcal{S}(n-1)$ such that $\tilde{\eta}_1 = \dfrac{N - \sum_{i=1}^{n-1} b_i}{\sum_{i=1}^{n-1} a_i}$ as defined in (*)

$$T(n-1) = \sum_{i=1}^{i=n} T_{s(i)} - T^0_{s(n)} + \sum_{i=1}^{i=n} T_{r(i)} - T^0_{r(n)} + \lfloor \tilde{\varrho} \rfloor$$

$$\tilde{\varrho} = T^{\tilde{\eta}_1}_{c(1)} - \sum_{i=2}^{n} T_{s(i)} - T^0_{s(n)} - T^{\tilde{\eta}_1-\eta_1}_{s(1)} = \varrho + T^{\tilde{\eta}_1-\eta_1}_{c(1)} + T^0_{s(n)} - T^{\tilde{\eta}_1-\eta_1}_{s(1)}$$

$$\eta_n = \sum_{i=1}^{n-1} \eta_i - \tilde{\eta}_i \,,\; T_{c(n)} < \sum_{i=1}^{n-1} T_{r(i)}$$

$$T(n) - T(n-1) = T^0_{s(n)} + T^0_{r(n)} - \lfloor \tilde{\varrho} \rfloor \,.$$

Case 1) $\tilde{\varrho} \le 0,\ T(n) - T(n-1) = T^0_{s(n)} + T^0_{r(n)} > 0$

Case 2) $\tilde{\varrho} > 0,\ T(n) - T(n-1) = T^0_{s(n)} + T^0_{r(n)} - \left(\varrho + T^{\tilde{\eta}_1-\eta_1}_{c(1)} + T^0_{s(n)} - T^{\tilde{\eta}_1-\eta_1}_{s(1)}\right)$

$$= T^0_{r(n)} - \left(\varrho + T^{\tilde{\eta}_1-\eta_1}_{c(1)} - T^{\tilde{\eta}_1-\eta_1}_{s(1)}\right) > 0 \text{ if } \varrho < T^0_{r(n)} + T^{\tilde{\eta}_1-\eta_1}_{s(1)} - T^{\tilde{\eta}_1-\eta_1}_{c(1)}$$

Effect of threshold, ς, and associated approximation $\tilde{\eta}_i$ on χ_i :

Define $\overline{\eta}_i = \operatorname{int}(\eta_i)$, $\tilde{\eta}_i = \overline{\eta}_i$ if $\eta_i - \overline{\eta}_i < \varsigma$, $\tilde{\eta}_i = \overline{\eta}_i + 1$ if $\eta_i - \overline{\eta}_i \ge \varsigma$

$$\chi_i \le \left\lfloor \gamma_i\phi_c\tilde{\eta}_{i-1} + \phi_s\,\tilde{\eta}_{i-1} + T^0{}_{s(i-1)} - \gamma_i\phi_c\tilde{\eta}_{i-1} - \phi_r\tilde{\eta}_{i-1} - T^0{}_{r(i-1)} \right\rfloor$$

$$\max(\chi)_i \le \gamma_i\phi_c(1-\varsigma) + \phi_s(1-\varsigma) - \gamma_i\phi_c\varsigma - \phi_r\varsigma$$

$$\max(\chi)_i \le T^1_{s(i)} + T^1_{c(i)} - \varsigma\left(T^1_{c(i)} + T^1_{s(i)} + T^1{}_{c(i-1)} + T^1{}_{r(i-1)}\right)$$

$$\max(\chi)_i \le T^1_{s(i)} + T^1_{c(i)} < T^1{}_{c(j)} + T^1{}_{c(1)} \,.$$

$\therefore$ thresholding will not violate condition 1 .

Effect of threshold, ς, and associated approximation $\tilde{\eta}_i$ on ω_i :

$$\omega_i = \lfloor T_{c(i-1)} + T_{r(i-1)} - \left(T_{s(i)} + T_{c(i)} \right) \rfloor$$

Define $\overline{\eta}_i = \operatorname{int}(\eta_i)$, $\tilde{\eta}_i = \overline{\eta}_i$ if $\eta_i - \overline{\eta}_i < \varsigma$, $\tilde{\eta}_i = \overline{\eta}_i + 1$ if $\eta_i - \overline{\eta}_i \ge \varsigma$

$$\omega_i = \left\lfloor \gamma_i\phi_c\tilde{\eta}_{i-1} + \phi_r\tilde{\eta}_{i-1} + T^0{}_{r(i-1)} - \gamma_i\phi_c\tilde{\eta}_i - \phi_s\tilde{\eta}_i - T^0{}_{s(i)} \right\rfloor$$

$$\max(\omega_i) = \gamma_i\phi_c(1-\varsigma) + \phi_r(1-\varsigma) - \gamma_i\phi_c\varsigma - \phi_s\varsigma$$

$$\max(\omega_i) = T^1{}_{c(i-1)} - \varsigma\, T^1{}_{c(i-1)} + T^1{}_{r(i-1)} - \varsigma\, T^1{}_{r(i-1)} - \varsigma\, T^1{}_{c(i)} - \varsigma\, T^1_{s(i)}$$

$$< T^1{}_{c(j-1)} + T^1{}_{r(j-1)} + T^1{}_{c(j)} + T^1{}_{s(j)} \,.$$

$\therefore$ thresholding will not violate condition 2 .
